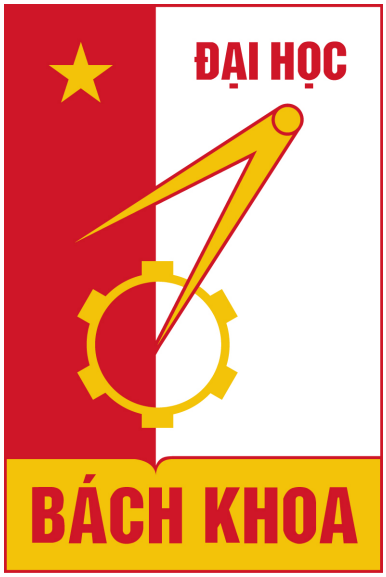
ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

o0o



Thiết kế thuật toán di truyền để giải quyết bài toán TSP

Môn học: Lập trình hướng đối tượng - IT3103

Giảng viên hướng dẫn: Nguyễn Thị Thu Trang

Mã lớp: 143577

Nhóm 03: Nguyễn Thị Minh Châu - 20214997
Nguyễn Thanh Nhật Bảo - 20210096
Võ Việt Bắc - 20205055
Nguyễn Minh Chiến - 20215000

Mục lục

Lời nói đầu

Lập trình hướng đối tượng (Object-oriented programming, viết tắt: OOP) là một mẫu hình lập trình dựa trên khái niệm "công nghệ đối tượng", mà trong đó, đối tượng chứa đựng các dữ liệu, trên các trường, thường được gọi là các thuộc tính; và mã nguồn, được tổ chức thành các phương thức.
Phương thức giúp cho đối tượng có thể truy xuất và hiệu chỉnh các trường dữ liệu của đối tượng khác, mà đối tượng hiện tại có tương tác (đối tượng được hỗ trợ các phương thức "this" hoặc "self"). Trong lập trình hướng đối tượng, chương trình máy tính được thiết kế bằng cách tách nó ra khỏi phạm vi các đối tượng tương tác với nhau. Ngôn ngữ lập trình hướng đối tượng khá đa dạng, phần lớn là các ngôn ngữ lập trình theo lớp, nghĩa là các đối tượng trong các ngôn ngữ này được xem như thực thể của một lớp, được dùng để định nghĩa một kiểu dữ liệu... OOP được xem là giúp tăng năng suất, đơn giản hóa độ phức tạp khi bảo trì cũng như mở rộng phần mềm bằng cách cho phép lập trình viên tập trung vào các đối tượng phần mềm ở bậc cao hơn.
Ngoài ra, nhiều người còn cho rằng OOP dễ tiếp thu hơn cho những người mới học về lập trình hơn là các phương pháp trước đó. Một cách giản lược, đây là khái niệm và là một nỗ lực nhằm giảm nhẹ các thao tác viết mã cho người lập trình, cho phép họ tạo ra các ứng dụng mà các yếu tố bên ngoài có thể tương tác với các chương trình đó giống như là tương tác với các đối tượng vật lý.
Hiểu được tầm quan trọng và sự hiệu quả của học phần OOP, nhóm được giao nhiệm vụ thực hiện bài tập nhóm nhằm nâng cao, nắm chắc kiến thức cũng như áp dụng vào vấn đề thực tiễn.
Cuối cùng, nhóm 3 xin gửi lời cảm ơn đến cô, anh trợ giảng cùng nhà trường đã tạo điều kiện cho chúng em được học tập và tiếp thu kiến thức bổ ích từ môn học. Xin chân thành cảm ơn !

Thành viên	Cụ thể
Nguyễn Thị Minh Châu (leader)	47%
Nguyễn Thanh Nhật Bảo	40%
Võ Việt Bắc	10%
Nguyễn Minh Chiến	3%

Bảng 1: Đóng góp

Chương 1

Khảo sát, đặc tả yêu cầu bài toán

1.1 Mô tả yêu cầu bài toán

Tổng quan: Thuật toán di truyền (GA) là một phương pháp tìm kiếm heuristic chịu ảnh hưởng của lý thuyết của Charles Darwin về sự tiến hóa tự nhiên. Quá trình chọn lọc tự nhiên được phản ánh trong GA, nơi những cá thể phù hợp nhất được được chọn lọc để sinh sản nhằm sinh ra con cháu thế hệ sau.

Bài toán Người bán hàng du lịch (TSP) là một bài toán nổi tiếng trong Khoa học Máy tính. Tuyên bố vấn đề khá đơn giản, cho một đồ thị G = (V, E), trong đó V là tập các nút, E là tập các cạnh, có thông tin bổ sung về khoảng cách giữa các nút. Mục tiêu của bài toán là đưa ra tuyến đường ngắn nhất bắt đầu từ V và kết thúc tại V, sao cho mỗi nút được thăm một lần. Dự án này triển khai một Thuật toán Gen (GA) để xác định tuyến đường tối ưu.

Trong dự án này, chúng tôi được yêu cầu giải quyết Bài toán Người bán hàng du lịch (TSP) bằng cách triển khai Thuật toán di truyền cùng với việc áp dụng các kỹ thuật OOP và Lập trình Java.

Yêu cầu kiến thức cơ bản: Thuật toán, Cấu trúc dữ liệu (không cần kiến thức nâng cao về thuật toán tiến hóa, có thể triển khai dự án này mà không cần kiến thức trước về GA)

Yêu cầu về giao diện người dùng (GUI):

- Thiết kế:
- Trên menu chính: tiêu đề của ứng dụng, 3 nút cho các thuật toán sắp xếp để người dùng giúp đỡ, giải quyết TSP và thoát
  - Menu Trợ giúp hiển thị cách sử dụng cơ bản và mục tiêu của chương trình

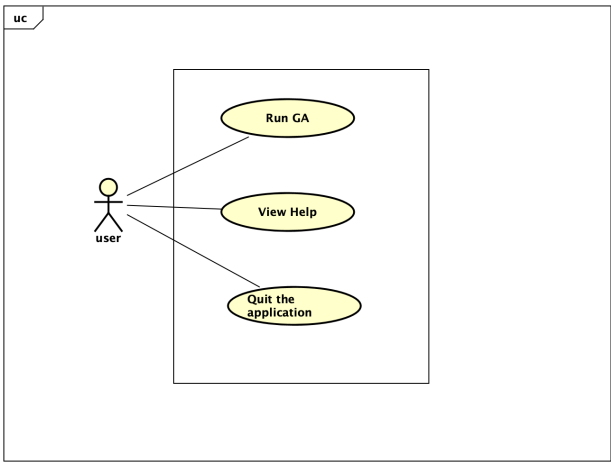
  - Tùy chọn Thoát thoát khỏi chương trình. Nhớ hỏi xác nhận

- Trong bản minh họa
  - Một nút để bắt đầu mô phỏng, cụ thể là "Bắt đầu"
  - Một hình minh họa cho cá thể hiện tại giữ fitness tốt nhất, trong cấu trúc mảng
  - Một hình minh họa cho cá thể hiện tại giữ fitness tốt nhất, trong cấu trúc đồ thị

1.2 Biểu đồ usecase

Giải thích:

- **View Help (Đại diện cho chức năng xem trợ giúp):** Người dùng có thể xem hướng dẫn sử dụng, giải thích các tham số và chức năng của ứng dụng. Người dùng cần nhấn vào nút Help.
- **Run GA (Đại diện cho chức năng chạy giải thuật di truyền):** Người dùng có thể sử dụng khi muốn hệ thống chạy thuật toán. Trường hợp sử dụng này yêu cầu người dùng nhập tham số (số lượng thành phố, kích thước quần thể và số lượng thế hệ), sau đó nhấn nút Run GA để bắt đầu chạy và tuyến đường tối ưu sẽ được in ra.
- **Quit the application (Đại diện cho chức năng thoát ứng dụng):** Người dùng có thể thoát khỏi ứng dụng một cách an toàn sau khi hoàn thành công việc hoặc không muốn tiếp tục bằng cách nhấn vào nút Quit.

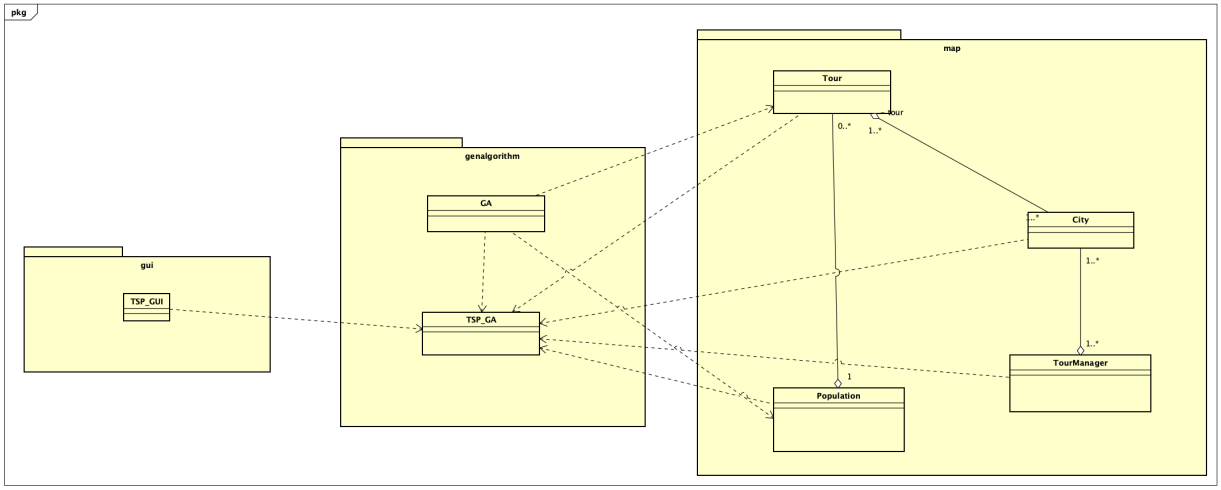


Hình 1.1: Biểu đồ usecase

Chương 2

Thiết kế hệ thống

2.1 Biểu đồ lớp tổng quan

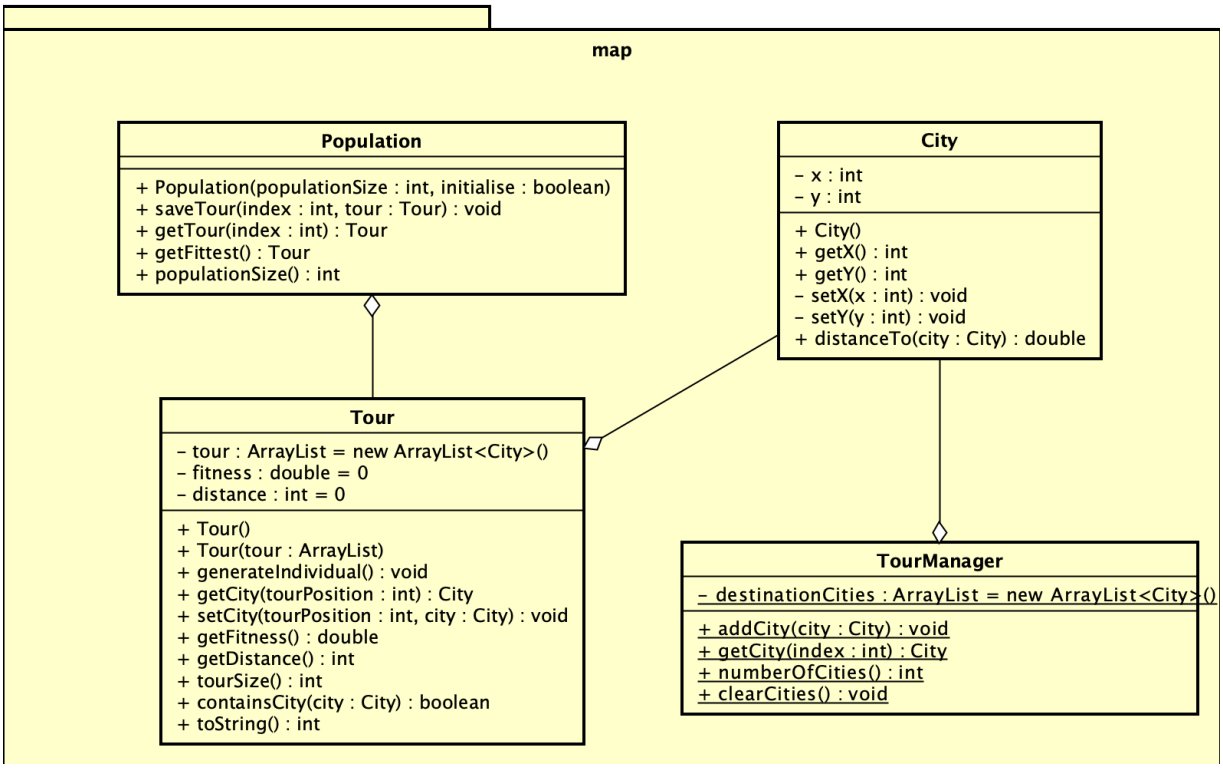


Hình 2.1: Biểu đồ gói

2.2 Biểu đồ lớp chi tiết

2.2.1 Package map:

- City: Đại diện cho một thành phố với tọa độ x và y, cung cấp các phương thức tính toán khoảng cách và truy cập tọa độ.
  - Thuộc tính: x và y: tọa độ của thành phố



Hình 2.2: Biểu đồ chi tiết package map

  - Phương thức:
    * City(): là 1 constructor, tạo 1 thành phố có tọa độ ngẫu nhiên
    * getX() và getY(): lấy giá trị x và y
    * setX(x: int) và setY(y: int): thiết lập giá trị cho thuộc tính x và y
    * distanceToCity(city: City): tính toán khoảng cách giữa hai thành phố

- TourManager: Lớp chứa tất cả các thành phố đích cho hành trình
  - Thuộc tính: destinationCities: một ArrayList lưu trữ các thành phố
  - Phương thức:
    * addCity(city: City): thêm một thành phố vào danh sách
    * getCity(index: int): lấy một thành phố từ danh sách dựa trên chỉ số index
    * numberOfCities(): lấy số lượng thành phố trong danh sách
    * clearCities(): xóa thành phố

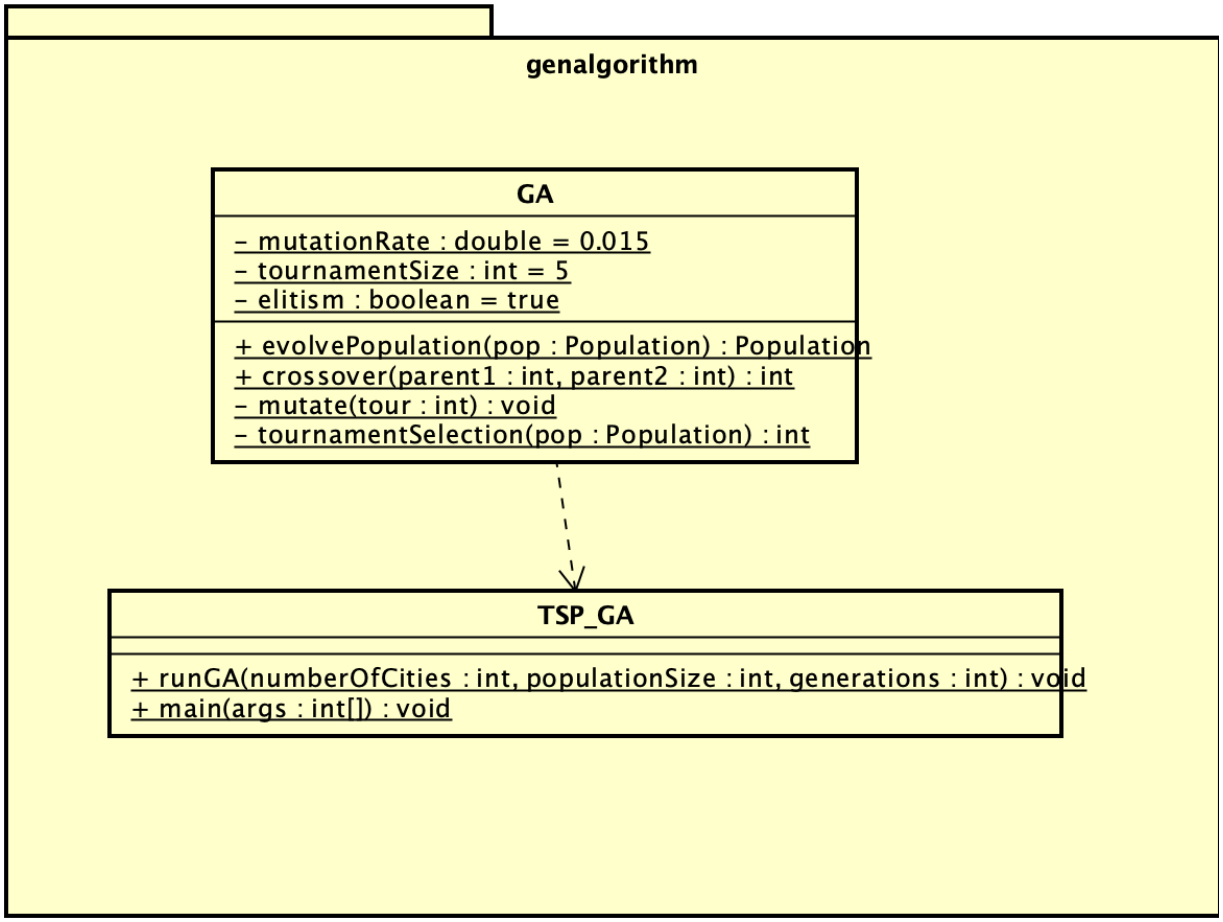
- Tour: Lớp mã hóa các tuyến đường, một mảng biểu diễn các thành phố được xếp theo thứ tự (1 cá thể)
  - Thuộc tính:
    * tour: một ArrayList lưu trữ các thành phố trong tour
    * fitness: Cache giá trị đánh hiệu suất của 1 cá thể tour
    * distance: Cache giá trị tổng khoảng cách
  - Phương thức:
    * Tour(): constructor tạo một tour trống với số lượng thành phố bằng số lượng thành phố trong TourManager
    * Tour(tour: ArrayList): Constructor có tham số để tạo một tour từ một ArrayList đã cho
    * generateIndividual(): tạo một tour ngẫu nhiên bằng cách thêm các thành phố từ TourManager vào Tour() rồi sử dụng shuffle để xáo trộn. Tạo cá thể ngẫu nhiên đảm bảo sự đa dạng của quần thể
    * getCity(tourPosition: int): lấy thành phố từ vị trí cụ thể trong Tour
    * setCity(tourPosition: int, city: City): Đặt một thành phố ở vị trí cụ thể trong Tour
    * getFitness(): lấy giá trị đánh giá hiệu suất của tour (nghịch đảo của tổng khoảng cách)
    * getDistance: lấy tổng khoảng cách
    * tourSize(): lấy số lượng thành phố trong tour
    * constainsCity(city: City): Kiểm tra xem tour có chứa thành phố hay không
    * toString(): trả về biểu diễn chuỗi của tour

- Population: Tập hợp các cá thể (quần thể)
  - Thuộc tính: tours: một mảng chứa các cá thể (tour) trong quần thể
  - Phương thức:
    * Population(populationSize: int, initialise: boolean): constructor tạo ra một quần thể với kích thước chỉ định, nếu initialise là true sẽ tạo và khởi tạo các cá thể bằng cách gọi generateIndividual()
    * saveTour(index: int, tour: Tour): lưu 1 cá thể tour vào mảng tours ở vị trí được chỉ định
    * getTour(index: int): lấy 1 cá thể tour ở mảng tours tại vị trí được chỉ định
    * getFittest(): lấy cá thể tour có độ thích nghi tốt nhất
    * populationSize: lấy kích thước quần thể

2.2.2 Package genalgorithm:

Chứa các classes liên quan đến thuật toán di truyền.

- GA: Thực hiện các phương thức liên quan đến quá trình tiến hóa như crossover, mutation và selection.
  - Tham số: mutationRate, tournamentSize, elitism là các tham số được khai báo Private static final với giá trị mặc định được đặt sẵn
  - Phương thức:
    * evolvePopulation(Population pop): Tiến hóa một quần thể qua một thế hệ. Điều này bao gồm việc tạo một quần thể mới, lựa chọn các cá thể tốt nhất từ quần thể hiện tại (có thể có chế độ elitism), thực hiện crossover và mutation để tạo ra các cá thể con mới.
    * crossover(Tour parent1, Tour parent2): Áp dụng phép lai giữa hai cá thể cha mẹ để tạo ra một cá thể con mới.
    * mutate(Tour tour): Đột biến một tour sử dụng phương pháp swap mutation.
    * tournamentSelection(Population pop): Lựa chọn một tour từ quần thể sử dụng phương pháp tournament selection.
- TSP_GA:
  - Phương thức:
    * runGA(int numberOfCities, int populationSize, int generations):
      · Nhận vào số lượng thành phố (numberOfCities), kích thước quần thể (populationSize), và số lượng thế hệ (generations).
      · Tạo ngẫu nhiên các thành phố và khởi tạo một quần thể ban đầu.



Hình 2.3: Biểu đồ chi tiết package genalgoritm

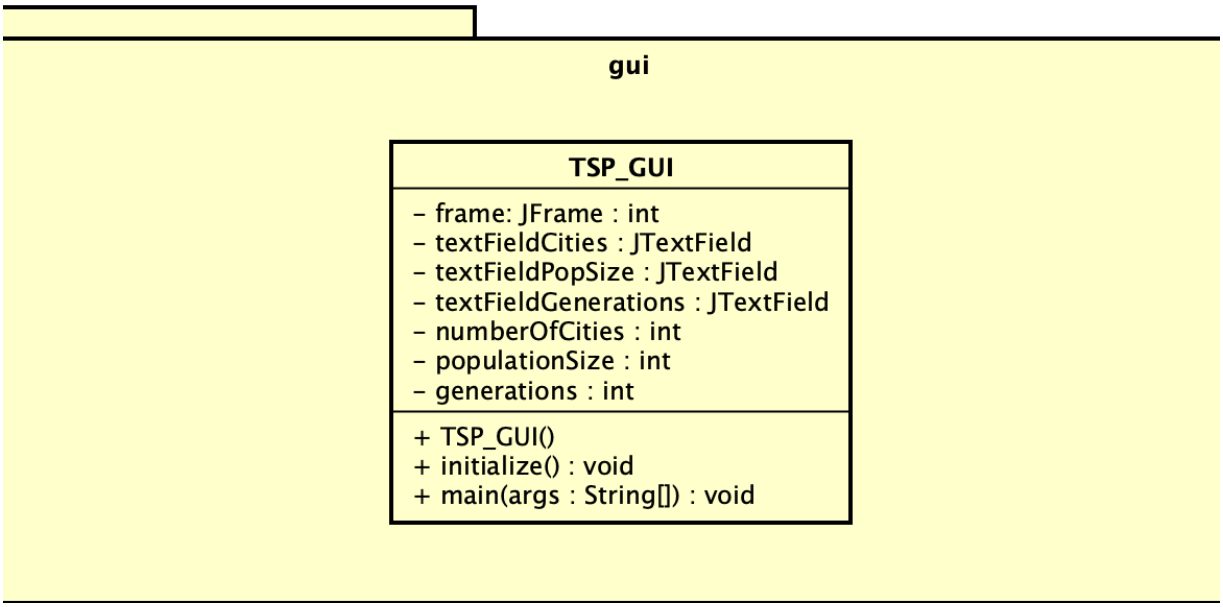
- In ra khoảng cách ban đầu giữa các thành phố trong tour tốt nhất.
- Tiến hóa quần thể qua số lượng thế hệ đã cho.
- In ra kết quả cuối cùng, bao gồm khoảng cách của tour tốt nhất và tour đó.

* Phương thức main(String[] args):
  - Nhận thông tin từ người dùng về số lượng thành phố, kích thước quần thể, và số lượng thế hệ thông qua bảng điều khiển.
  - Gọi phương thức runGA với thông tin đã nhập để thực hiện giải thuật di truyền.

2.2.3 Package gui:

Chứa các classes và components liên quan đến giao diện người dùng (GUI).

- TSP_GUI: Class chính quản lý giao diện, sự kiện và tương tác giữa thuật toán và người dùng.
- CityPanel: JPanel tùy chỉnh để vẽ thành phố và các đường nối trên GUI.



Hình 2.4: Biểu đồ chi tiết package gui

Chương 3

Các kỹ thuật lập trình hướng đối tượng đã áp dụng

1. Tính đóng gói: Các class như City, Tour, và Population có các thuộc tính được đặt private, và các phương thức getter và setter được cung cấp để truy cập và cập nhật dữ liệu. Điều này giúp bảo vệ dữ liệu khỏi sự truy cập trực tiếp từ bên ngoài.

2. Tính kế thừa: Class TSP_GUI được xây dựng trên cơ sở của class JFrame. Điều này thể hiện quan hệ "is-a", trong đó TSP_GUI là một loại JFrame. Class Population là một tập hợp (collection) của các tour, thể hiện mối quan hệ "has-a" với class Tour. Điều này thể hiện tính kế thừa của Population từ Tour và mô phỏng mối quan hệ "is-a".

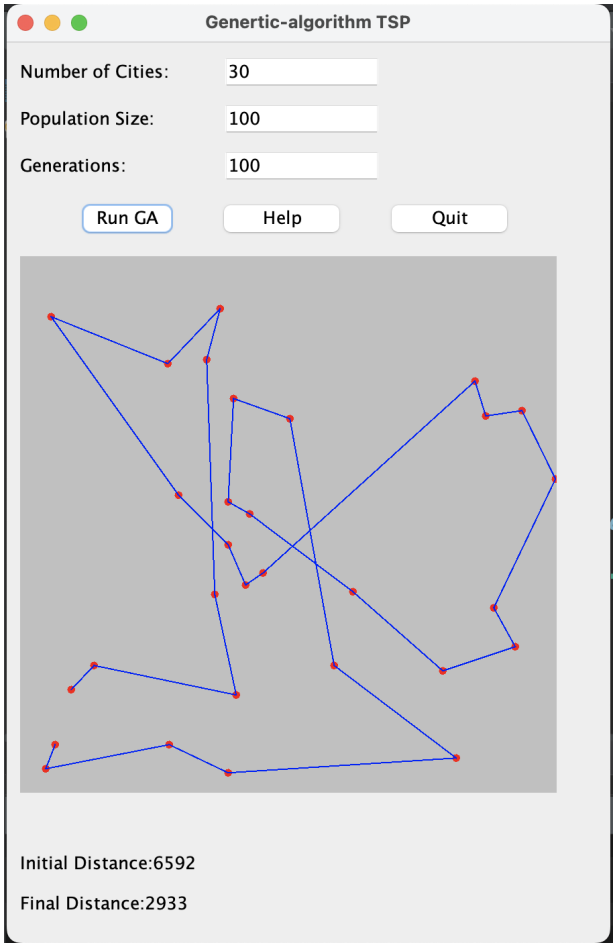
3. Tính đa hình: Sử dụng Timer trong GUI để tạo đa nhiệm, cập nhật giao diện đồ họa sau mỗi thế hệ của thuật toán mà không làm đóng băng giao diện người dùng.

4. Tính hợp thành: Class TSP_GUI sử dụng sự hợp thành để tích hợp các thành phần như JLabel, JTextField, và JButton. Điều này giúp xây dựng giao diện người dùng bằng cách kết hợp các thành phần nhỏ thành một cấu trúc lớn và dễ quản lý. Class Population chứa một mảng (tours) của các đối tượng Tour, thể hiện mối quan hệ "has-a" thông qua sự hợp thành. Class Population tự nó là một đối tượng độc lập có khả năng duy trì một tập hợp các tours.

Chương 4

Demo chương trình

Video demo



Hình 4.1: GUI

Chương 5

Phân công công việc cụ thể

1. Nguyễn Thị Minh Châu 20214997
    - Tạo lớp City, lớp Tour, lớp TourManagement, lớp Poppulation, lớp GA
    - Làm slide
    - Viết báo cáo
2. Nguyễn Thanh Nhật Bảo 20210096
    - Vẽ general class diagram, detailed class diagram
    - Tạo lớp TSP_GUI, lớp TSP_GA
3. Võ Việt Bắc 20205055
    - Vẽ general class diagram, usecase diagram
4. Nguyễn Minh Chiến 20215000

Chương 6

Kết luận

Dự án bài tập lớn lập trình hướng đối tượng đã cung cấp cho sinh viên cái nhìn tổng quan về cách một dự án hoạt động với phương pháp lập trình hướng đối tượng, cùng với đó là cách quản lý và hoạt động nhóm khi tham gia hoặc tổ chức một dự án công nghệ như thế nào. Sau chương trình học cùng với bài tập lớn, sinh viên chúng em đã có nền tảng kiến thức vững chắc về lập trình hướng đối tượng, một trong những kĩ năng vô cùng quan trọng trong công việc sau này dù có phát triển theo bất kì định hướng nào. Ngoài ra nhũng kĩ năng mềm về khả năng làm việc với git, khả năng trao đổi thảo luận nhóm cũng được cải thiện và phát triển rất nhiều

Chương 7

Tài liệu tham khảo

Tham khảo triển khai thuật toán: https://www.theprojectspot.com
Tham khảo thiết kế GUI: https://www.youtube.com/watch?v=94p5NUogClM
Thư viện swing: https://zetcode.com/javaswing/